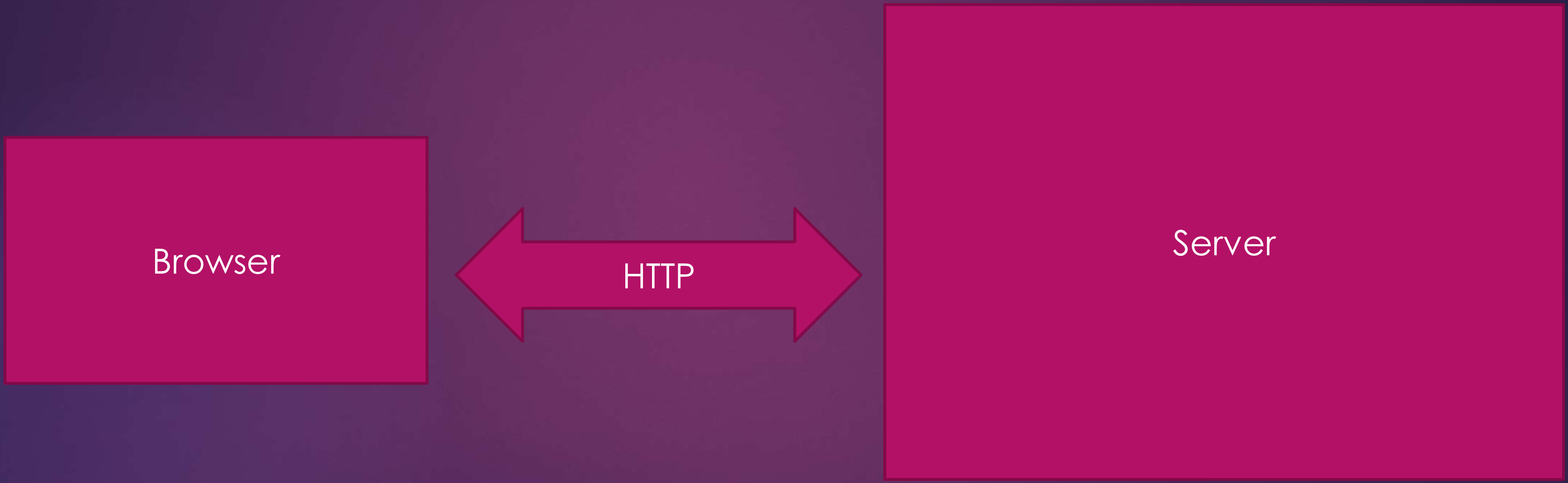
Model View Controller

MVC FRAMEWORK, IN GENERAL.

HTTP Request/Response



PHP Processing on the Server

- What happens in the server black box?

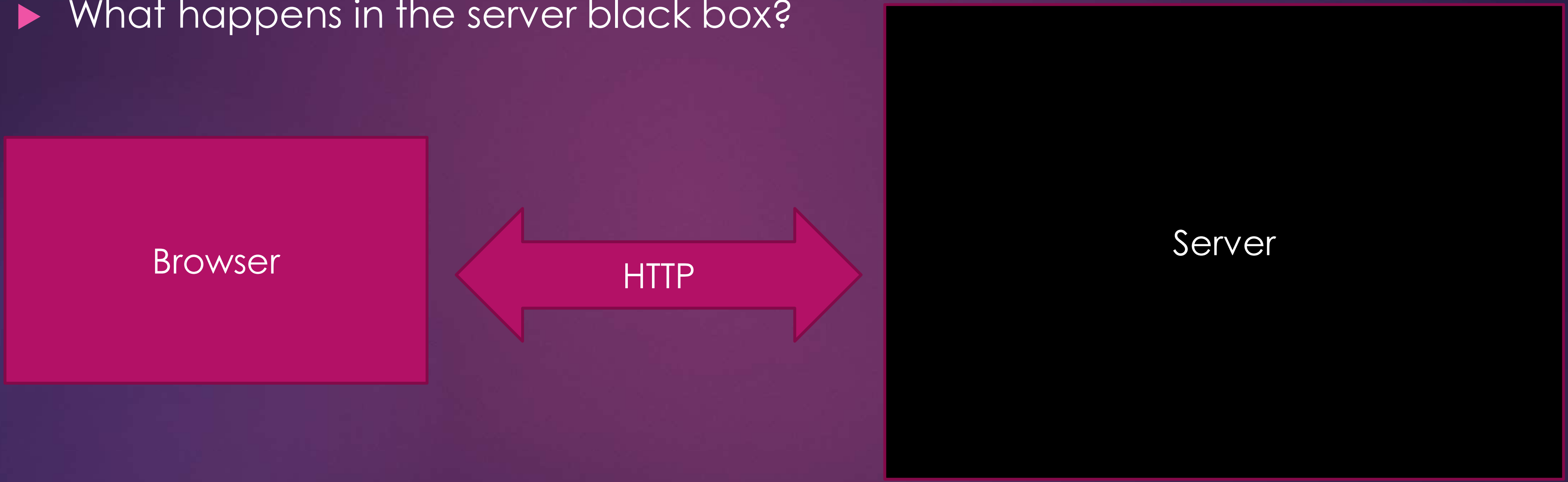
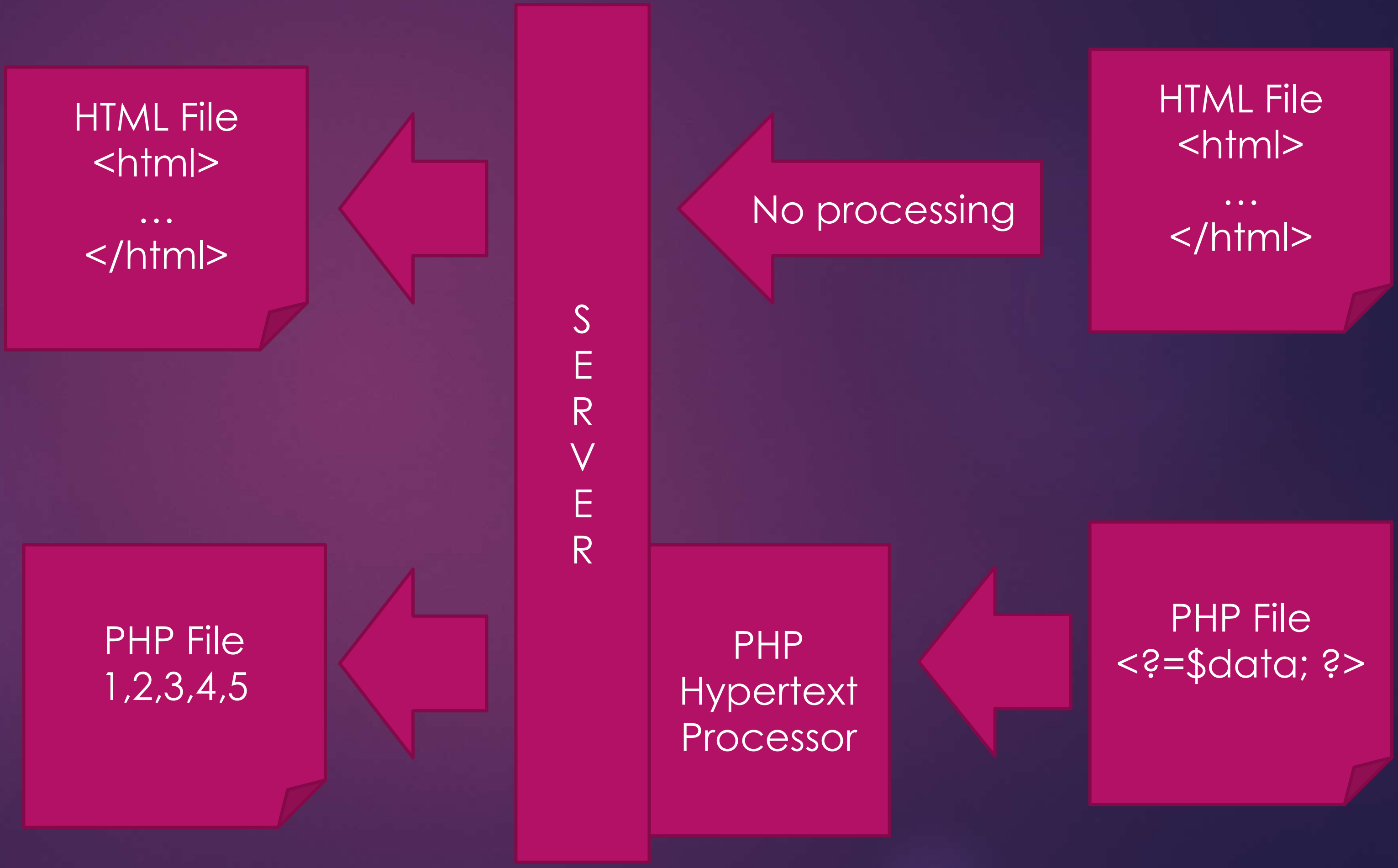


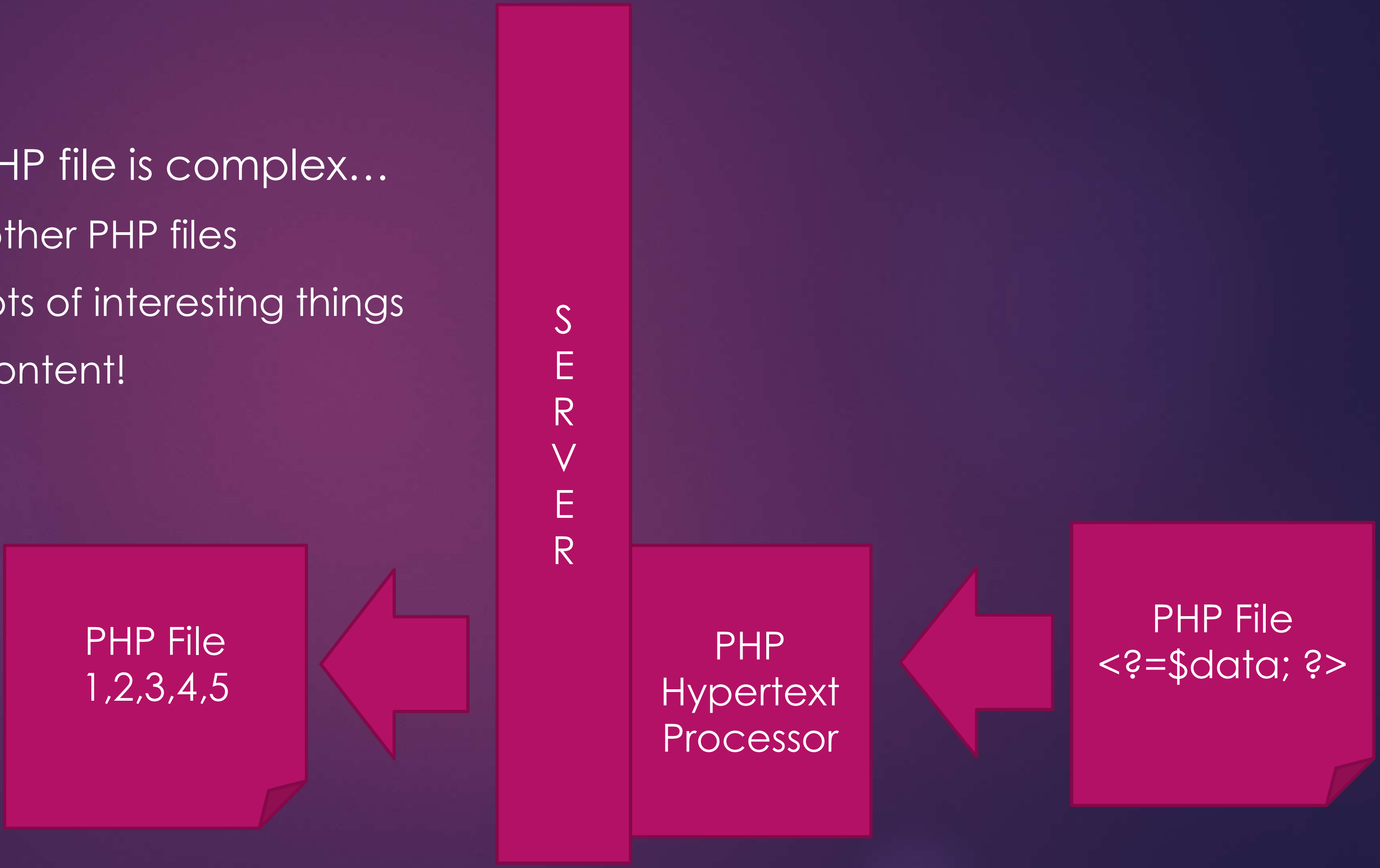
Diagram of overall flow, client/server

- PHP Files are processed through the PHP Hypertext Processor
- Static HTML files are not.



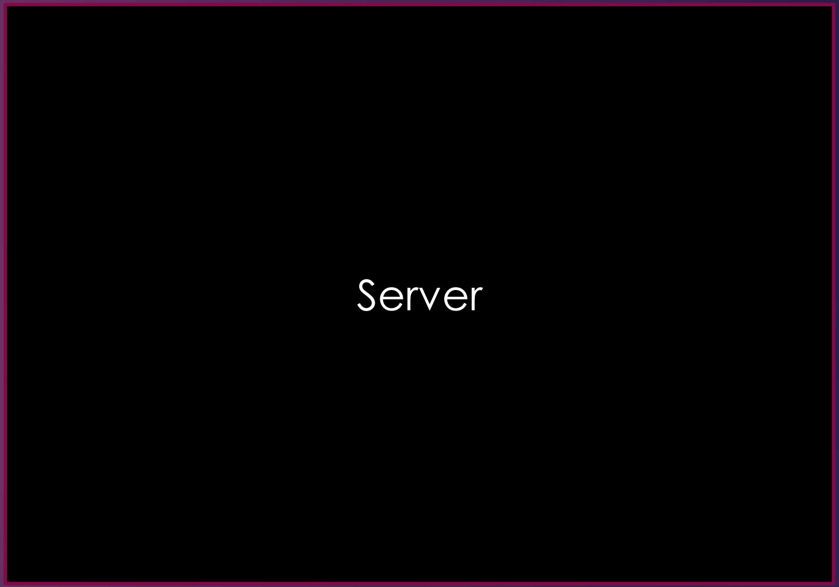
Now, what happens on the server?

- But what if that PHP file is complex…
  - It can include other PHP files
  - Those can do lots of interesting things
  - Yay dynamic content!



Inside that black box…

- Request comes in
- Server processes the request
  - Sets up state
  - Gets the data to use
  - Processes data
  - Formats results
  - Lays out results
- Page is returned to the browser
- Browser renders the page



Set up State

- Specific code that knows where the state is and what format it is in.
- Parses out the URL, parameters, etc…
- Includes the desired other classes that might be used
  - Though not many, most of the logic in one file

Get Data to Use

- Grabs the data, knowing:
  - Exactly where the data are stored
  - Exactly how the data are stored
  - Opens files, connects to DB, etc…
  - Maybe even put some data right in the code!

Process Data

- <!DOCTYPE html>
- <?php
  - //All of the above…
  - //plus
  - //lots of code to process data
  - (several lines later)
  - <html>
    - …

Format Results

- Not much to do here, variables are already declared local to the file.
- Hopefully, we didn't make too big of a mess.
- We needed to make variables for everything
  - Not so much automatic getting by calling a method
  - Maybe stored in an object, but not necessarily

Lay out results

- <html>
- …
- <div class="productWidget">
  - …
- </div>

- Even If in a loop, the code lives in potentially multiple places.

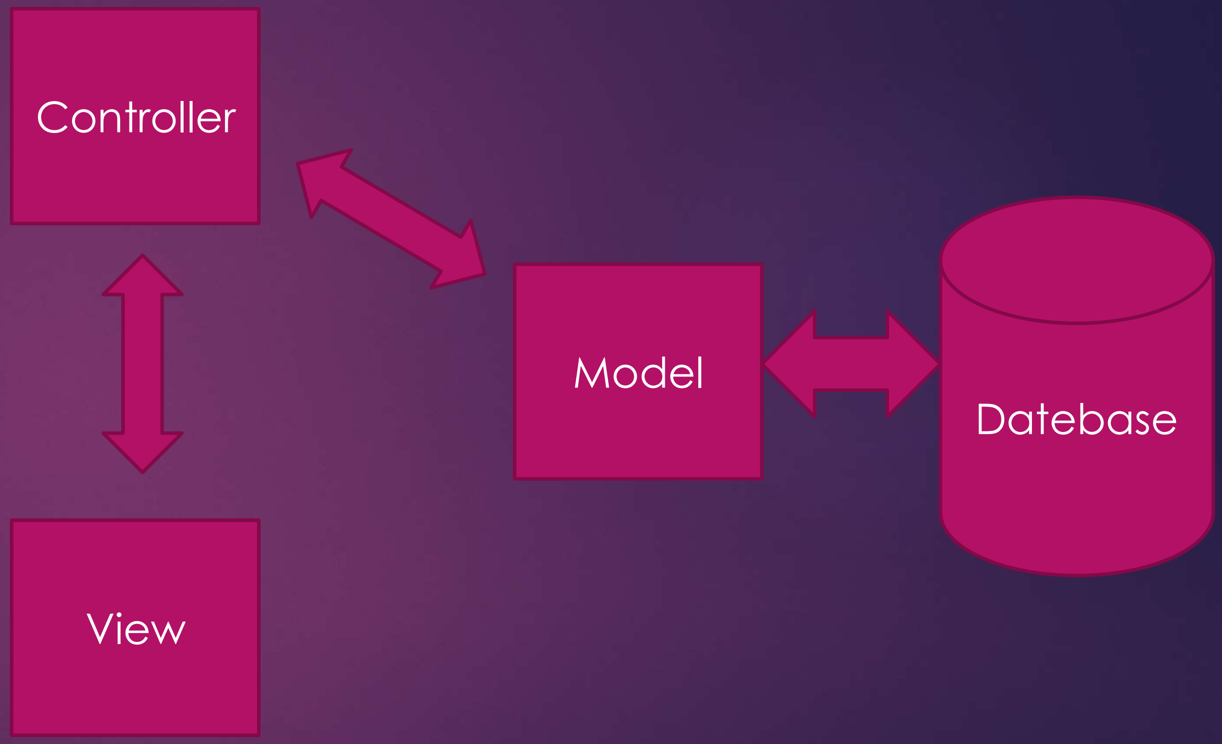
Doing this with MVC

Back to our flow…

- Sets up state
- Gets the data to use
- Processes data
- Formats results
- Lays out results

With MVC

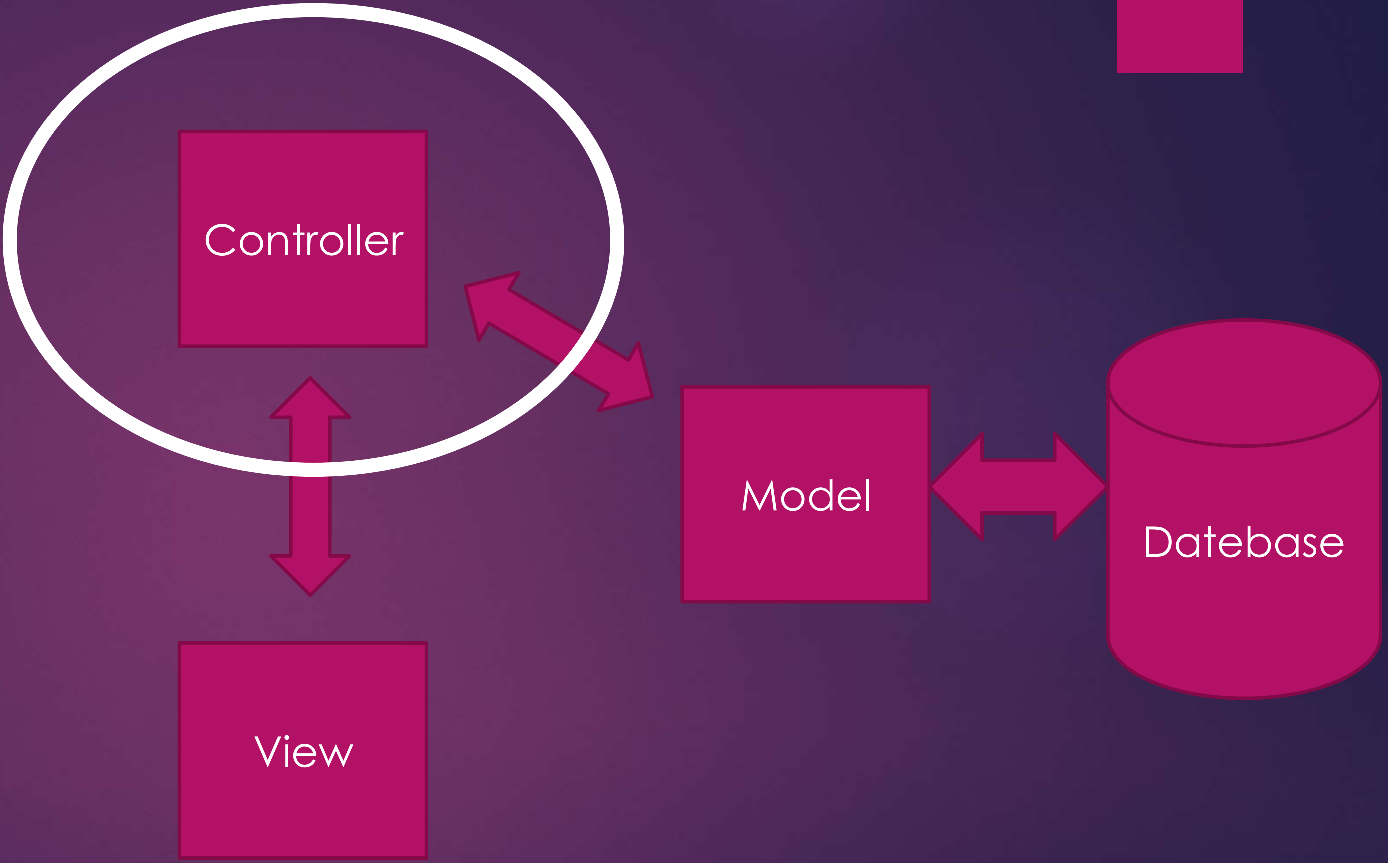
- Sets up state
- Gets the data to use
- Processes data
- Formats results
- Lays out results



Set up State

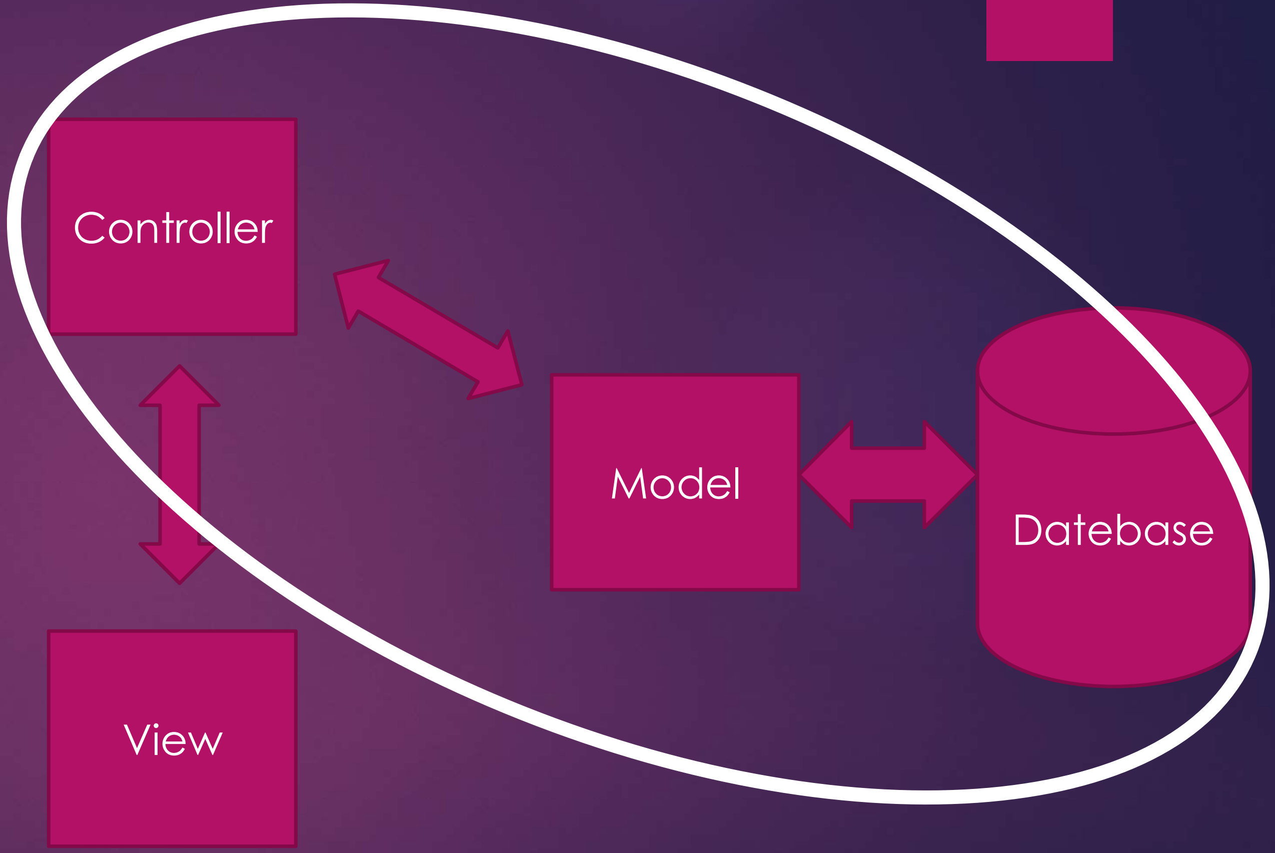
- Grab state
- Get context
- Start processing

- The controller knowns what it needs to accomplish the task



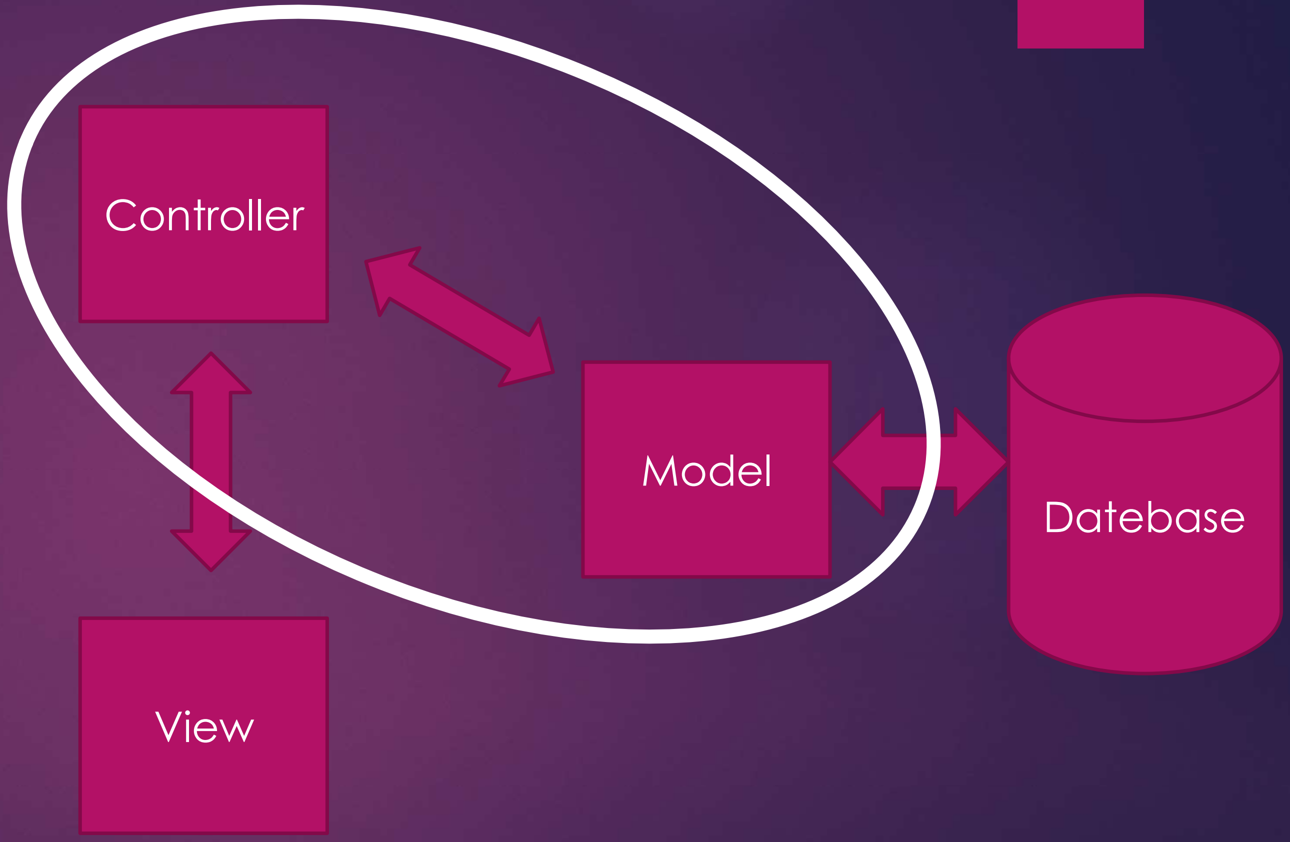
Get Data to Use

- The controller knows which data it needs.
- The model knows how to get that data.
- The model links to the DB
- Does the controller care what the DB looks like?



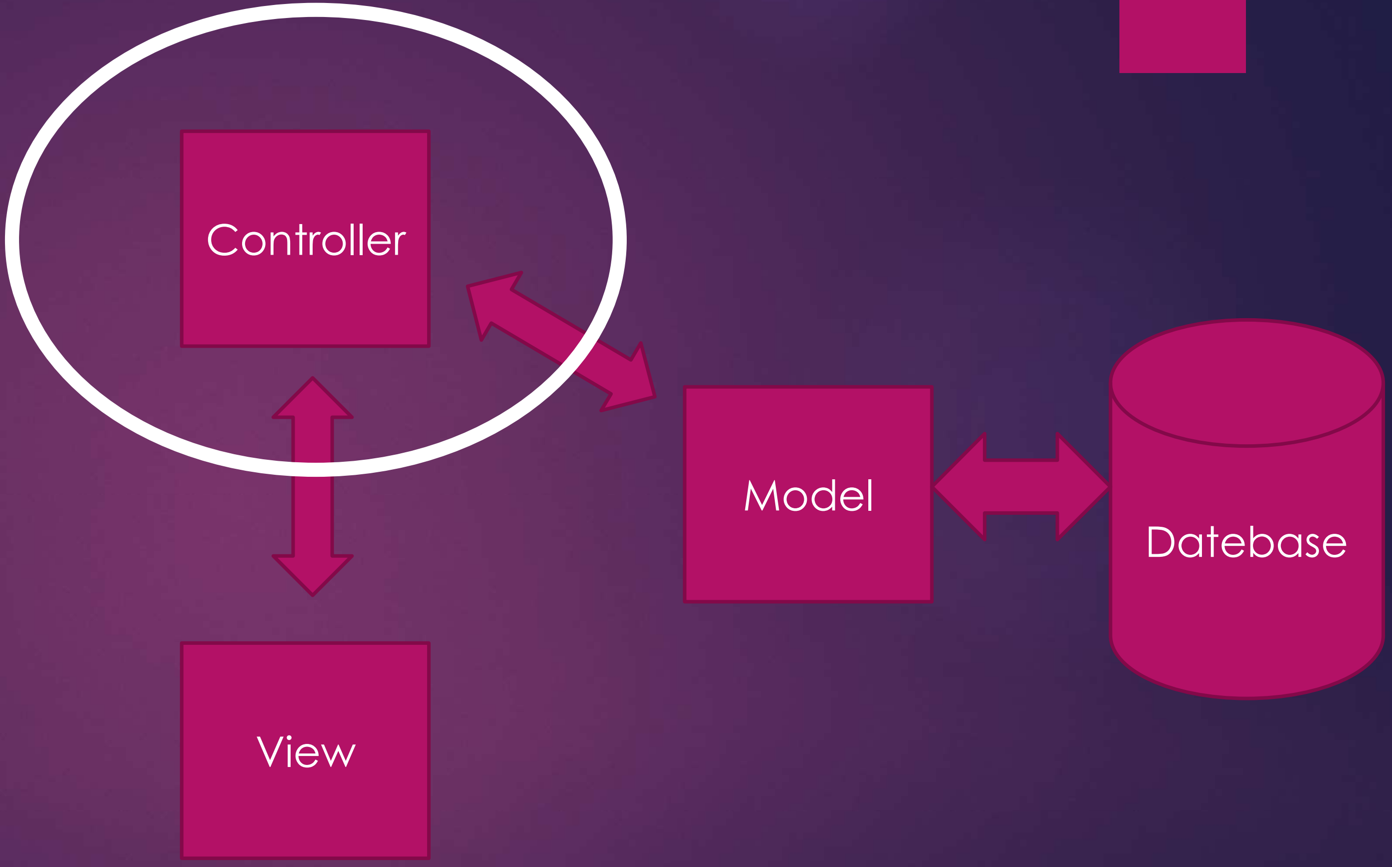
Process Data

- Processing of Data can happen two places, the Model or the Controller
- Depends on what is being processed.
- Is the processing specific to the Data, then Model.
- Is the processing specific to the action, then Controller



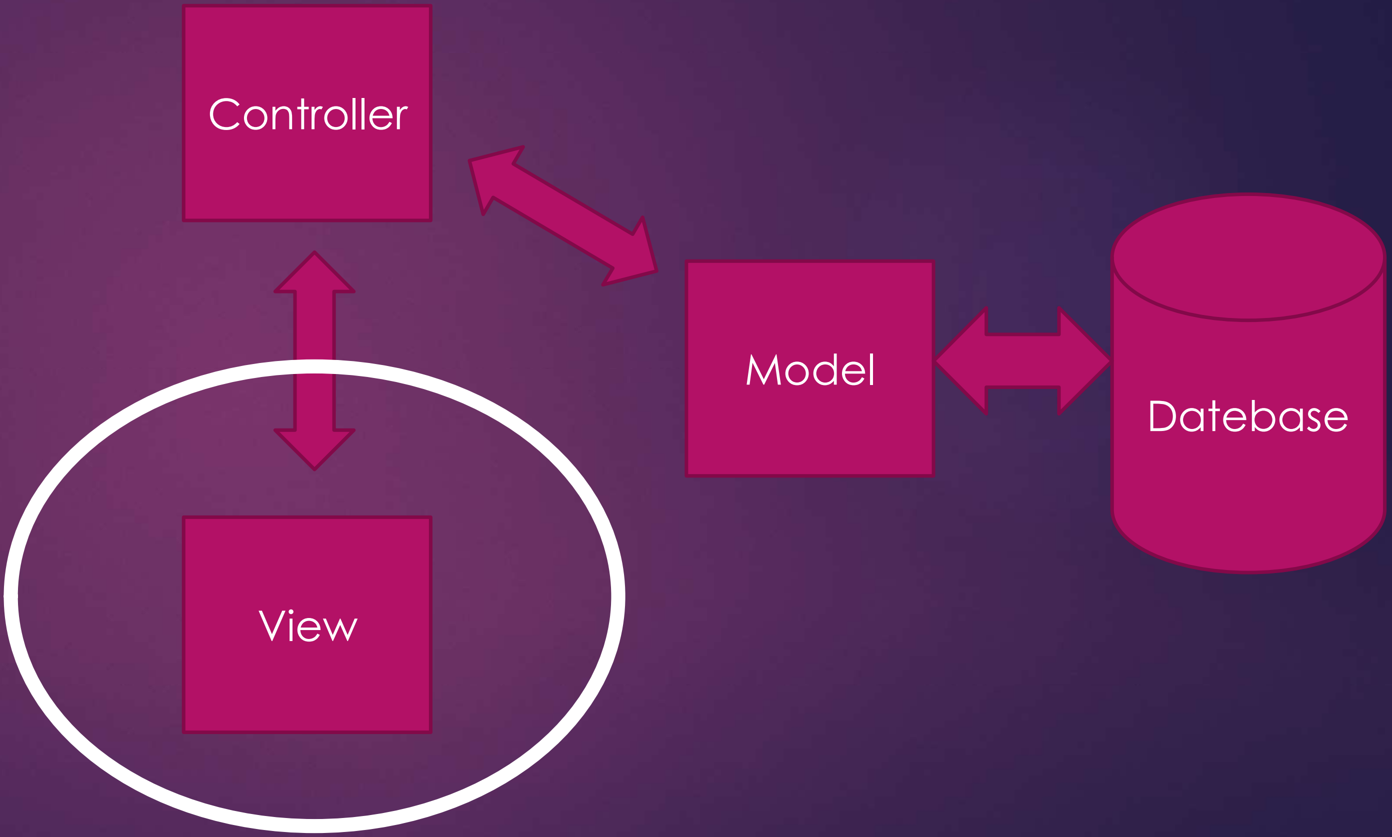
Format Result

- Once the data is processed, it is made into a format to be read by the View.
- Remember the earlier lecture… Lots of formats to choose from!
- Arrays of Model objects are very common.

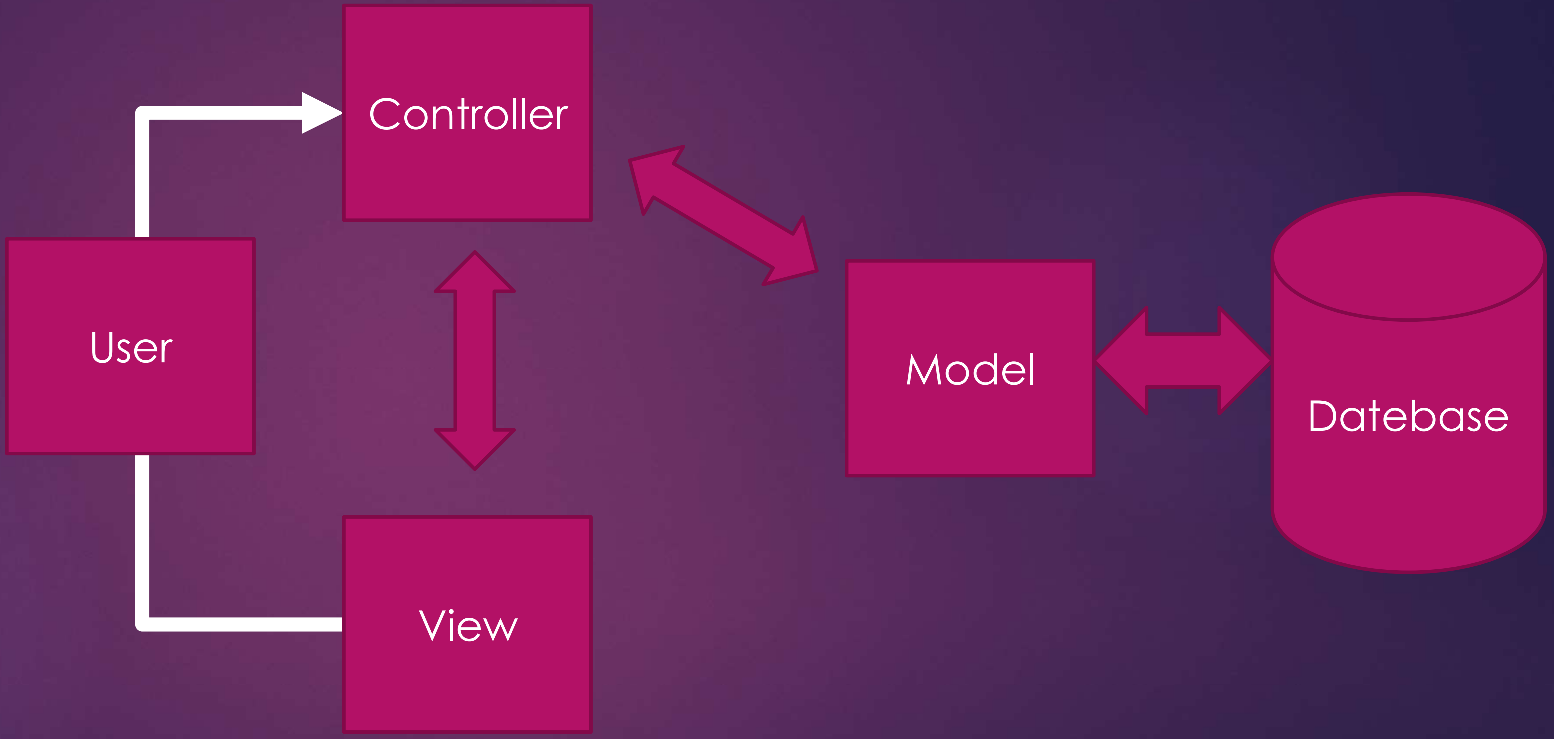


Lay Out Results

- How is the data presented to the user?
- The Model doesn’t care.
- The Controller doesn’t care.
- Only the View cares!
- The view knows the format it expects and how the data should look.



Then back to the controller…



More on Controllers

- Where to keep the business logic?
  - Controller?
    - Very specific to a certain page in the application
    - What if we want that logic to be “universal”?

More on Controllers

- Code re-use using components
  - Enables code re-use between models
  - Really just a class, we can import and use
  - Almost a model, but doesn’t connect to the DB
    - Might use models, though, to do this.

More on Models

- Supported data sources…
  - Files
  - Formal Databases (MySQL, for example)
  - Other web services (NCBI, for example)
  - Anywhere else you can think of!

More on Models

- Models often use ORM
  - Object Relational Mapping
- Each instance represents one entry of that model’s type of data
- Create the model, make changes:
  - Save those changes
  - Delete the record
- Give the record to a view
  - The work has been done, treat the model like a property list

More on Views

- Views can easily be swapped out for other ways to present the same data.
- We don’t want to process the data again or change that logic
- Different data visualizations can often tell a different story with the same data

More on Views

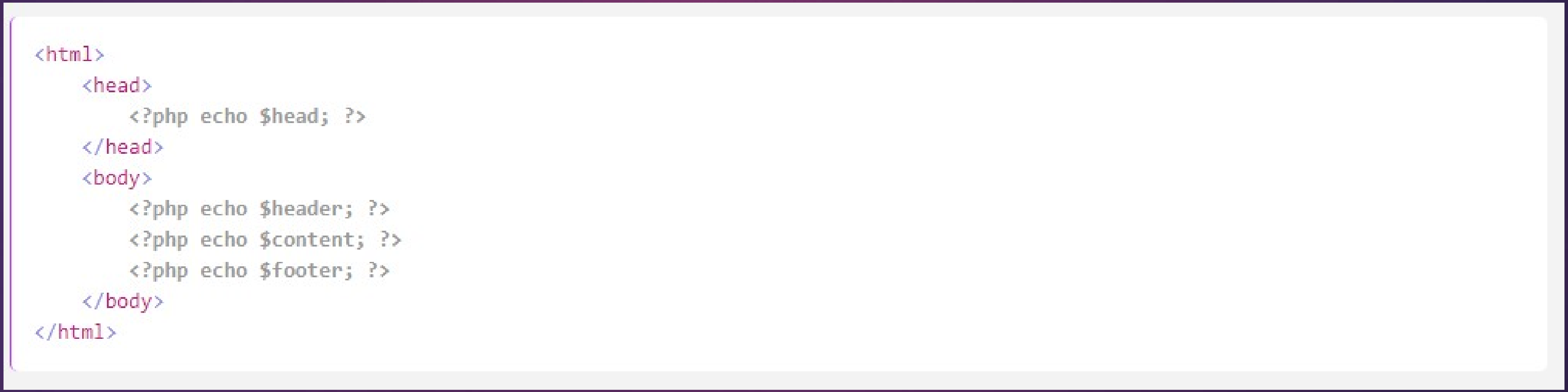
- Views can call other views
  - Makes a view a type of “widget”
  - Highly useful for things that repeat on a page.
  - Can call from within a loop
    - Maybe looping over models
    - Maybe looping over User IDs
    - Maybe looping over whatever you want!

Other useful pieces…

- Layouts & Templates
- Presenters
- Routers

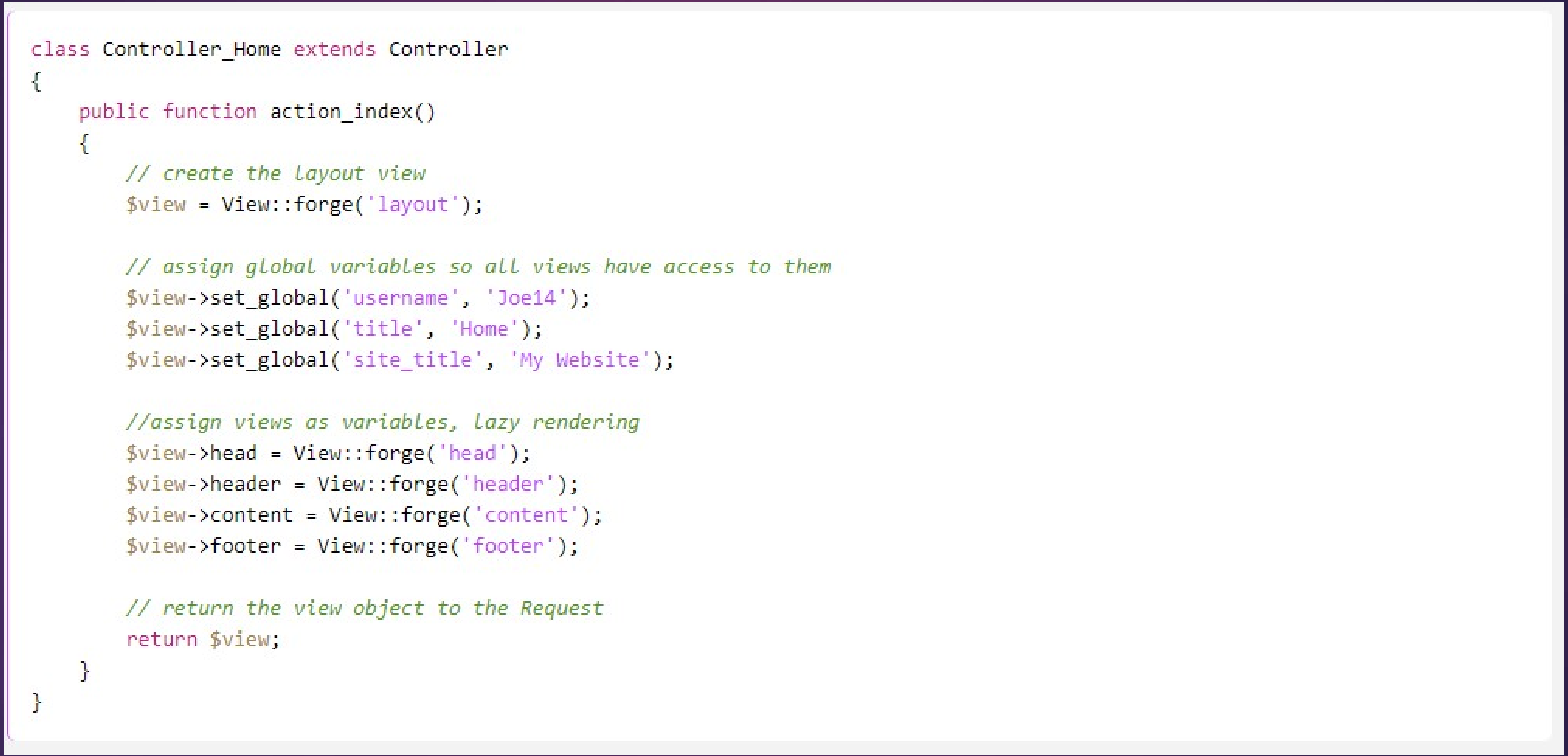
Layouts

- We've seen including a header/footer…
- What about:



```
<html>
    <head>
        <?php echo $head; ?>
    </head>
    <body>
        <?php echo $header; ?>
        <?php echo $content; ?>
        <?php echo $footer; ?>
    </body>
</html>
```

The downside…



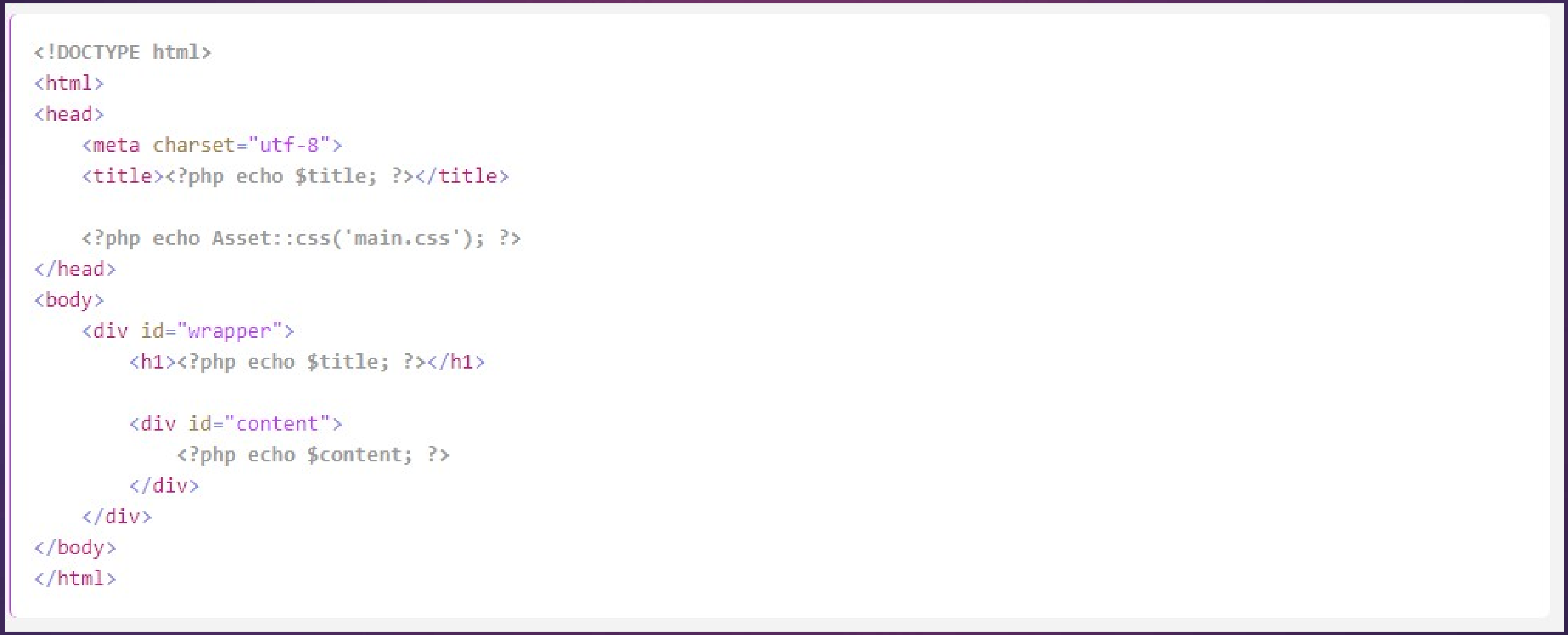
```php
class Controller_Home extends Controller
{
    public function action_index()
    {
        // create the layout view
        $view = View::forge('layout');

        // assign global variables so all views have access to them
        $view->set_global('username', 'Joe14');
        $view->set_global('title', 'Home');
        $view->set_global('site_title', 'My Website');

        //assign views as variables, lazy rendering
        $view->head = View::forge('head');
        $view->header = View::forge('header');
        $view->content = View::forge('content');
        $view->footer = View::forge('footer');

        // return the view object to the Request
        return $view;
    }
}
```

Much Better:



```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title><?php echo $title; ?></title>

    <?php echo Asset::css('main.css'); ?>
</head>
<body>
    <div id="wrapper">
        <h1><?php echo $title; ?></h1>

        <div id="content">
            <?php echo $content; ?>
        </div>
    </div>
</body>
</html>
```

Presenters

- A combination of Controllers and Actions
  - Thinks “Elements”
  - Amazon uses these a LOT
  - …So do lots of other large sites
    - News sites
    - Facebook
    - Twitter
    - …and integration for all of the above.

Routers

- Translate a URL into which controller to call
  - We'll look at how Fuel PHP Handles this…
  - Benefit:
    - From the URL, you know exactly which controller and method was called
    - From the URL, you know exactly which view file was called
    - Nice!
- Don't have to worry about directory structure when generating URLs
- Can easily link between pages
  - Also, it's intuitive to the programmer AND the user!